# Comp 4603

# Advanced C++

|  |  |  |  |
| --- | --- | --- | --- |
| Assignment | 9 | Part | 2 |

Student Name:Alisher Shamayev

BCIT ID: A01182685

In this assignment, you will examine template and inheritance through the lens of mathematics.

Background

In 1924, a Polish mathematician named Jan Łukasiewicz invented something called Polish notation. It was

refined in the early 1960s by Edsger Dijkstra who developed Reverse Polish Notation to take advantage of

the Stack data structure.

Binary infix operators are called binary because they require two operands. They are called infix because

the operator is placed **between** the operands:

2 + 2 = 4

4 - 2 = 2

5 / 3 = 1 (assuming we are using ints)

Reverse Polish Notation (RPN) is a mathematical notation where operators follow their operands. Instead

of using a binary infix operator, RPN uses a binary postfix operator:

2 2 + = 4

4 2 - = 2

5 3 / = 1

Consider our usual notation. When we mix our operations using binary infix operators, wemust implement

rules of precedence. The use of parentheses can result in dramatically different results:

2 - 3 \* 4 = -10

(2 - 3) \* 4 = -4

RPN (postfix) notation removes the need for parentheses! RPN’s greatest advantage is clear when we

consider expressions that contain more than one operand. If we want an operation to take precedence,

we just put the operator immediately to the right of the two operands:

2 3 4 \* - = -10

2 3 - 4 \* = -4

RPN doesn’t just eliminate parentheses and keystrokes. It’s flexible. If we use a Stack by pushing operands

onto the Stack until we reached an operator, we can pop operands off the Stack, transform them with the

operator, and push the result back onto the Stack as we go, letting us calculate complex partial results

without having to save them in multiple locations.

That’s exactly what we’re going to do. We’ll use a bit of inheritance and abstraction to create a hierarchy

of operations. Then we’ll build an RPNCalculator class that contains just a few methods.

Ordinarily for a project like this, we would use as many existing classes as possible. Why re-invent the

wheel, right? Ordinarily we would use the Stack class.

Ordinarily.

But not for this assignment. For this assignment, in order to gain some experience with simple data structures

and exceptions, you will do something extraordinary. You will implement your own Stack.

1 Submission requirements

This assignment must be submitted in Learning Hub before the due.

Late submissions will **NOT** be accepted.

2 Grading scheme

Your assignment will be marked out of 10.

[1 point] submitting your project on time

[2 points] project pass compile

[1 point] loop prompts

[2 points] be able to handle basic valid input (up-to 3 operands with 2 operators)

[2 points] be able to handle complex valid input (up-to 10 operands with 9 operators)

[2 points] be able to handle invalid input – error message & not crashing

3 Implementation requirements

Implement the following:

(1) You need to build a Reverse Polish Notation calculator. The RPNCalculator will keep prompting user for inputs.

For valid input, your program must respond by printing the expression inside a set of square brackets followed by the result:

[1 2 3 4 5 6 7 8 9 10 + + + + + + + + +] = 55

For invalid input, an error message will be shown as “input is invalid. Please try again”.

You will place the main() method inside a class called RPNCalculator.

(2) Be able to handle invalid input (not crash)